Systems programming

9 – Concurrency and parallelism

MEEC LEEC MEAer LEAer MEIC-A

João Nuno Silva

TÉCNICO LISBOA

Bibliography

- Time-Sharing Computer Systems, John McCarthy, MIT, 1964
- McCarthy, John, et al. "THOR: a display based time sharing system.", Proceedings of the April 18-20, 1967, spring joint computer conference. 1967.
- Bauer, Walter F. "Computer design from the programmer's viewpoint.", Papers and discussions presented at the December 3-5, 1958, eastern joint computer conference: Modern computers: objectives, designs, applications. 1958.
- Concurrent Programming for Scalable Web Architectures, Benjamin Erb, Section 2.3
- Principles of Concurrent Programming, M. Ben-Ari, Chapter 1
- Operating Systems – design and Implementation, Section 2.4

Why Concurrency

- So far most or all of your study in computing has assumed
  - One thing happened at a time
- Called sequential programming
  - everything part of one sequence

Concurrency

- Concurrency
  - Execution of two or more tasks overlap in time
  - Purpose
    - decrease response time
  - mechanism:
    - switch between different threads of control
    - work on one thread when it can make useful progress; when it can't, suspend it and work on another thread

Concurrency

- In the real world different activities often proceed at the same time .
  - But inter-related (one activity affect the others)
  - They are concurrent
- Computers are concurrent
  - Multiple resources
    - dozen processors, 10 disks, and 4 network interfaces; I/O devices
    - screen, keyboard, mouse, camera, microphone, speaker, network interface
  - Multiple users/applications

Concurrency

- Relevant to the programmer
  - Network services need to be able to handle multiple requests from their client
  - Most applications today have user interfaces while simultaneously executing application logic
  - Parallel programs need to be able to map work onto multiple processors
  - Need to mask the latency of disk and network operations

First computers

- Batch processing computers
  - First computers
  - Slow, one task at the time
  - Transistors, magnetic core memory

- 1945 – ENIAC
  - 18000 vacuum tubes
  - 7200 crystal diodes
  - 1,500 relays
  - 70,000 resistors
  - 10,000 capacitors,
  - 150 kW, 27 t
  - 100 word memory, 5000 SUMs per second
- 1957 - IBM 709
  - 100 kW, 1 Ton
  - 32K words memory, 42000 SUMs per second

Time sharing WF Bauer (1958)

- Laboratory computer
  - sequential operations + slow data access
- Buffered compute
  - sequential operations + fast data access
- Automatic computer
  - Operations + concurrent data access
- Parallel system
  - current operation of large components
- Self organizing System
  - Periodically examine its own operations and auto matically take certain steps to change its function.

Time sharing WF Bauer (1958)

- The central idea here is that each large metropolitan area would have one or more of these super computers.
- The computers would handle a number of problems concurrently.
  - Input-output equipment installed on their own premises
  - would buy time on the computer
    - the same way we buy power and water from utility companies.
- Tasks are scheduled by a supervisor
  - Each work has a priority

Time sharing John McCarthy (1964)

- I should like to go on now to consider how the private computer can be achieved.
  - It is done by time sharing a large computer.
  - Each user has a console that is connected to the computer by a wired channel such as a telephone line.
  - The consoles are two kinds,
    - one cheap and the other better but more expensive (the cheap console is simply an electric typewriter that is use for both input and output.
    - The more expensive console can include a cnathode-ray-tube unit on which the computer can display pictures and text

Time sharing John McCarthy (1964)

- Operation of such a system as it appears to the user of the typewriter console.
  - When the user wants service, he simply starts typing in a message requesting the service.
  - The computer is always ready to pay attention to any key that he may strike, just as the telephone system is always ready for youu to lift the receiver of the hook.
  - As soon as the key is depressed on the typewriter, a signal is sent to the computer.
  - The effect of the signal is to make the computer interrupt the pro after the current instruction has been executed and jump temporarily to a program that determines what the typewriter wants.

Time sharing John McCarthy (1964)

- At any moment some of the typewriters
  - will be inactive
  - some will be in the middle of entering messages into the computer,
  - some will be in the process of typing out characters
  - some will be in a status of wanting programs run.
- This is one example of a time-sharing system.
  - There are a large number of users.
  - Each user has his own console, gets service from the computer whenever he desires it, and has the computer maintain his files for him.

Time sharing Requirements

- Large primary memory
- Interruption system to handle errors as well as input and output
- Completely nonstop operation
- Erroneous programs must be prevented from damaging other programs
  - it must interrupt if the program attempts any memory references outside its allotted region,
  - programs that get into endless loops must be prevented from wasting computer time.
- Advances memory management (relocation/virtual)
- Secondary storage enough to maintain users’ files.

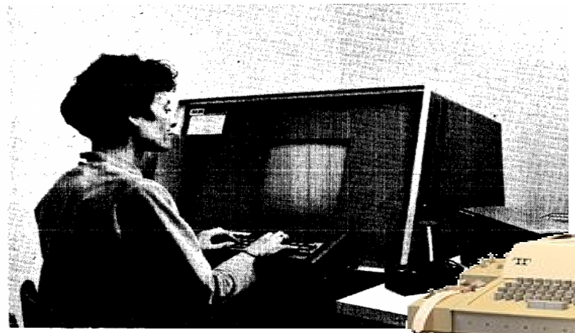
Time-sharing application

- BBN time-sharing system
  - increase the effectiveness of the PDP-1 computer
  - for applications involving man-machine interaction
- Allowing the five users
  - each at his own typewriter to interact with the computer
  - just as if he had a computer all to himself.

- 1966 - Thor on PDP-1
  - 8K Words
  - 22x4Kw drum memory
  - 5 users programs
  - Each user
    - Sees 4kwords
    - Response < 1s

Time-sharing application

- THOR 1966
  - Control computer based teaching laboratories
  - Text editing
  - General purpose programming
  - Better interaction CRT vs teletype
  - Understand trade-offs



- PDP-1
  - 4K Words
  - US$ 120K
  - 20 user programs
  - 28 consoles

Time sharing Computers

- Time-sharing Computers
  - Multiple users
  - Single CPU
  - Transistors

- 1966 - Thor on PDP-1
  - 4K Words
  - US$120K
- IBM 7090 (Super IBM 709)
  - 6x faster 3x cheaper
- 1964 - IBM S/360 Model 67
  - 512 Kbyte
- 1966 – DEC PDP-7
  - 4K words

Computer history

- Multitasking/Interactive Operating Systems
  - multiple-programs
  - Single CPU
  - Word and excel running at the same time
- Multiple tasks inside the same program
  - Excel, loading a file while updating a table

Concurrency

- Doing many things during the same interval
  - Not necessary at the same time
- Solve/run multiple units of tasks
  - with a single shared resource
  - during a period of time
- On computers, time and CPU are shared
  - At one instance only one task is executing in the CPU
- Gives the user
  - The illusion of parallelism

Concurrency

- In the real world different activities happen at the same time
  - Are inter-related (one activity affect the others)
  - Are concurrent
- Computers are concurrent
  - Multiple shared resources
    - Processors used by various programs
    - Disks being accessed by various programs
    - Network interfaces being used by various programs
  - Multiple users/applications

Concurrency

- Relevant to the programmer
  - Network services need to handle multiple requests
  - Most applications today have user interfaces while simultaneously executing application logic
  - Allows masking the latency of disk and network
- Concurrent programs automatically become parallel
  - If multiple processors exists

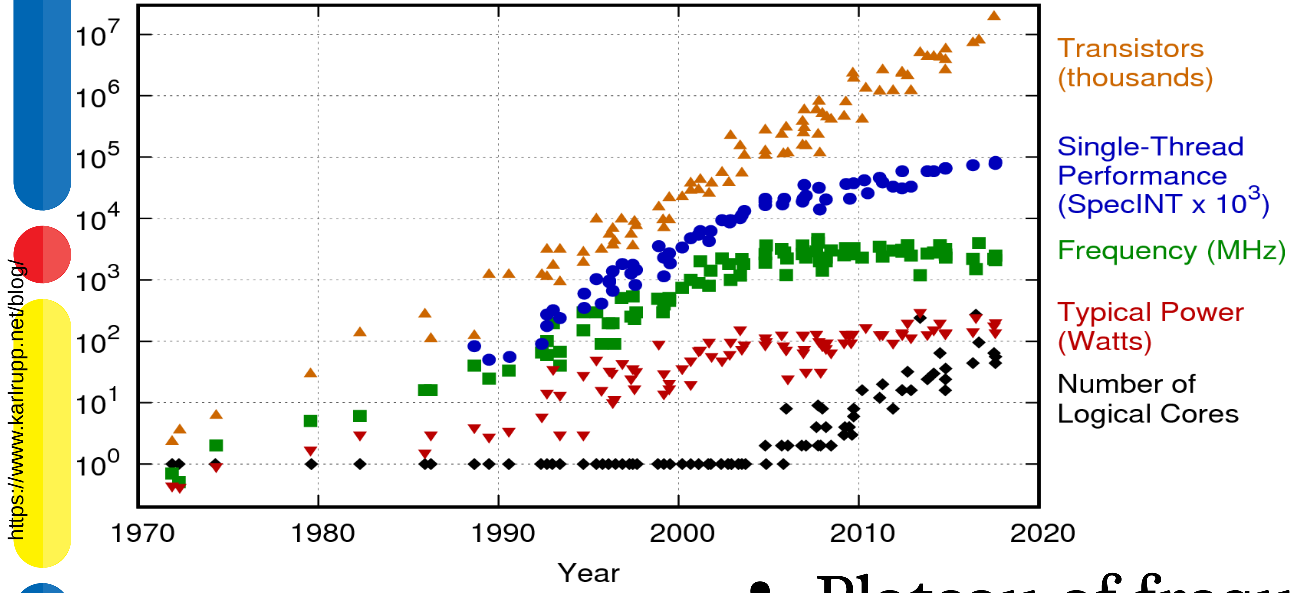
What comes after Concurrency?

- On our desktop....
- From roughly 1980-2005 desktop computers
  - got exponentially faster at running sequential programs
  - About twice as fast every couple years
- But nobody knows how to continue this
  - Increasing clock rate generates too much heat
  - Relative cost of memory access is too high

What comes after Concurrency?

- But transistors were getting smaller
- But we can not keep making “wires exponentially smaller”
  - Moore’s “Law”
- so put multiple processors on the same chip
  - “multi-core”

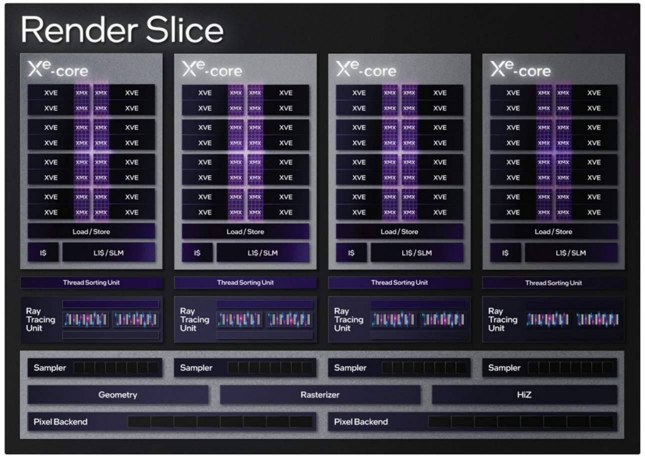
What comes after Concurrency?



- Plateau of frequency
- More transistors inside each chips
- More cores inside each chip

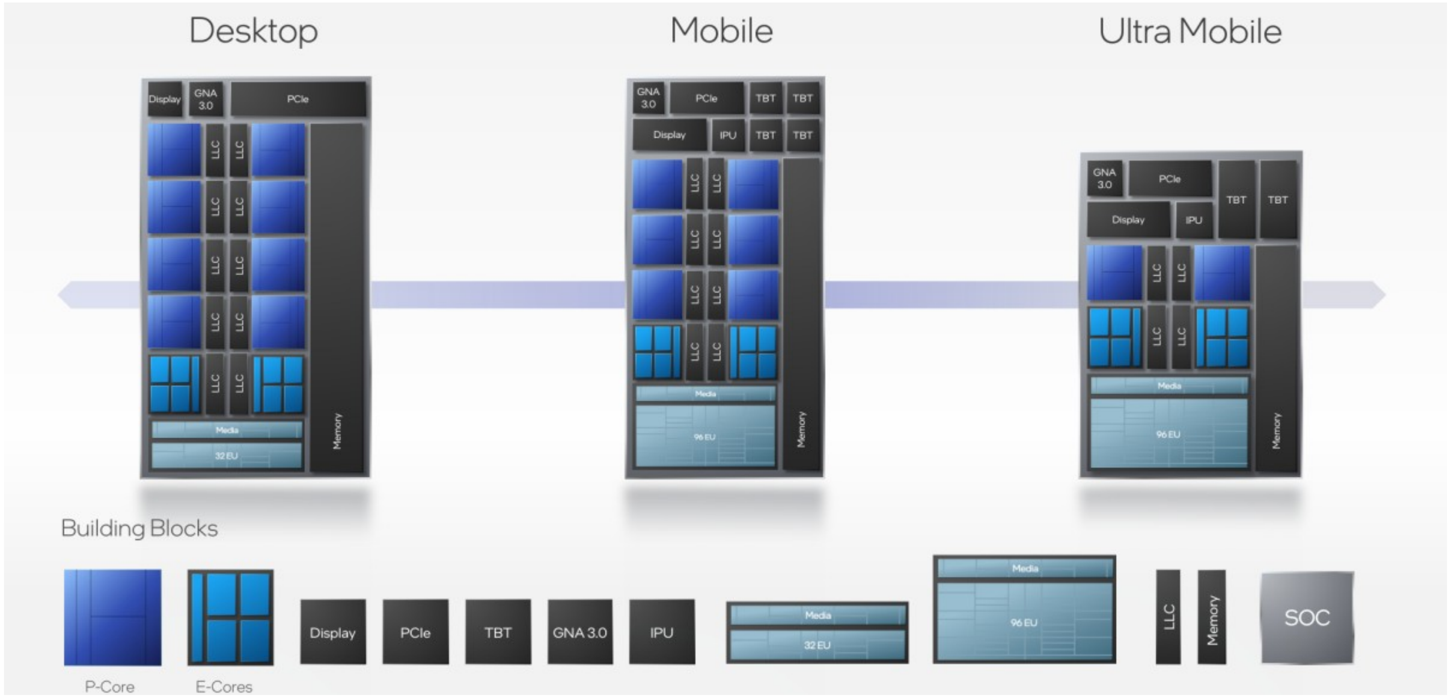
… we get to parallelism

- All computers have 4 / 8/16/… processors
  - Wait a few years and it will be 32, 64, 128
  - The chip companies have decided to do this not a “law”
- We have GPUs…
  - Intel Arc A380 $139
    - 8 Xe core
    - 1024 GPU Cores
    - 128 MXM Engines



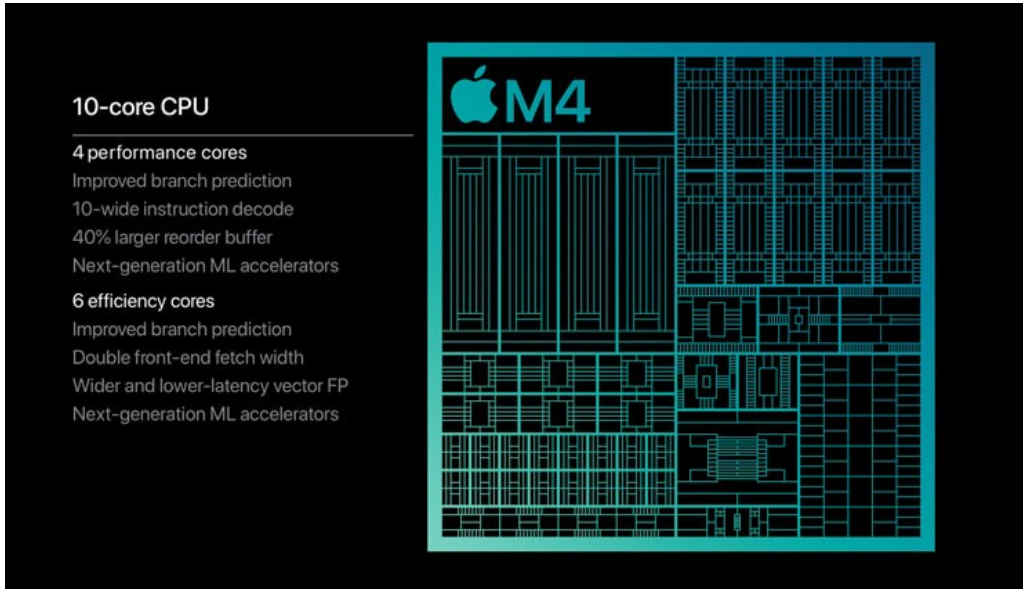
… we get to parallelism

- Intel - P-CORES and E-CORES



… we get to parallelism

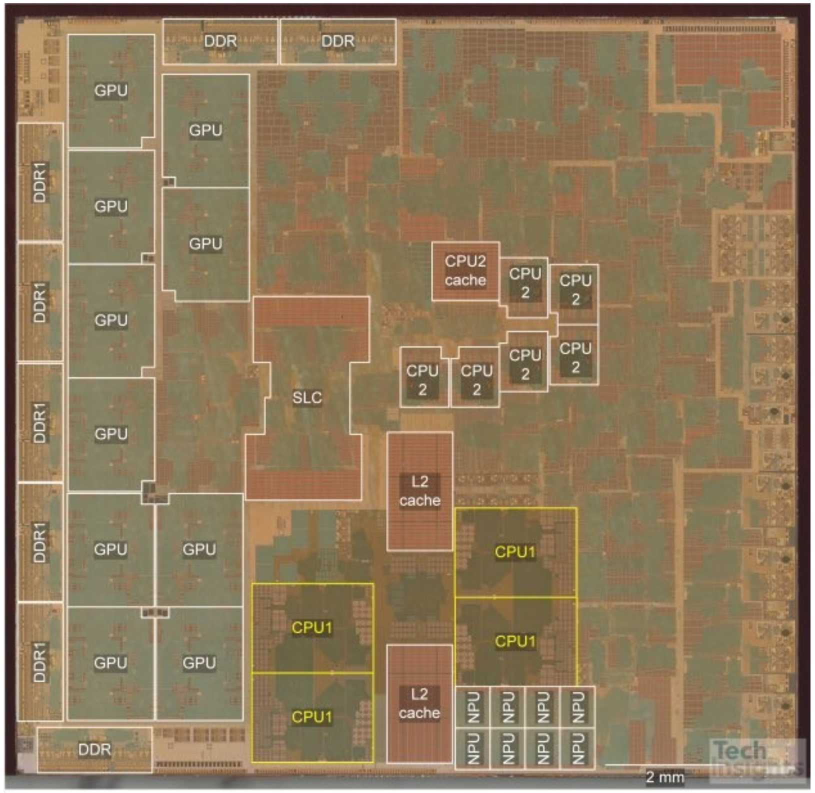
- Apple



10-core CPU

4 performance cores
Improved branch prediction
10-wide instruction decode
40% larger reorder buffer
Next-generation ML accelerators

6 efficiency cores
Improved branch prediction
Double front-end fetch width
Wider and lower-latency vector FP
Next-generation ML accelerators



Parallel computers

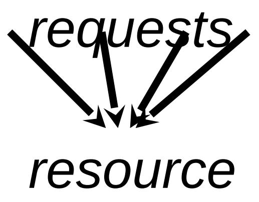
- What can you do with them?
  - Run multiple totally different programs at the same time
    - Already do that? Yes, but with time-sharing
- Do multiple things at once in one program
  - Requires rethinking everything
    - from asymptotic complexity
    - to how to implement data-structure operations
- Split complex tasks into simpler independent ones

Parallelism

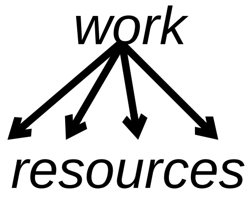
- Doing many things at the same time instead of sequentially
- Simultaneously
  - Requires multiple processing units

Parallelism vs. Concurrency

- Concurrency:
  - Correctly and efficiently manage simultaneous access to shared resources



- Parallelism:
  - Use various independent resources to solve a problem faster



- There is some connection:
  - Both use threads
- If parallel computations need access to shared resources
  - then the concurrency needs to be managed

Parallelism vs. Concurrency

- Perform many tasks simultaneously
- Purpose
  - improves throughput
- Mechanism:
  - many independent computing devices
- Decrease run time of program by utilizing multiple cores or computers
  - running your web crawler on a cluster versus one machine.

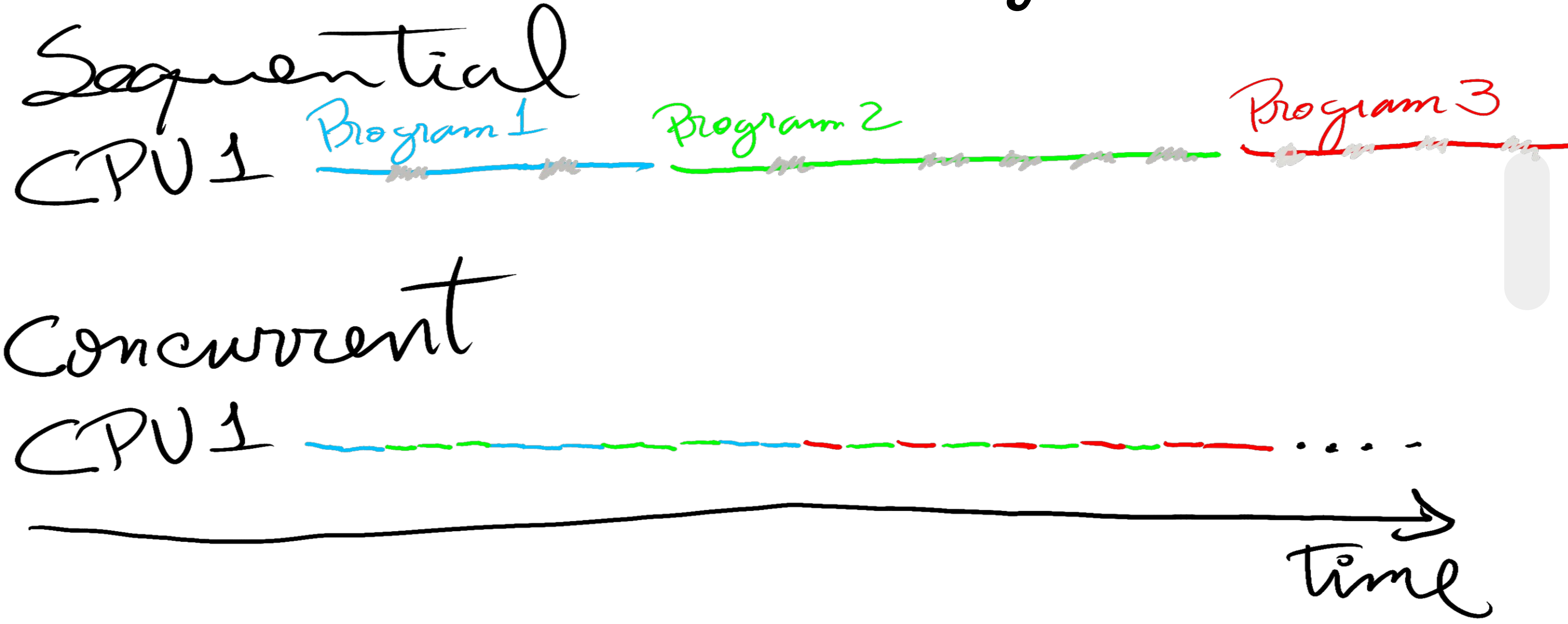
Parallelism vs **Concurrency**

- In each instance only one tasks is running
- Purpose
  - decrease response time
- Only uses one core or computers
  - But user thinks various cores are being used simultaneous

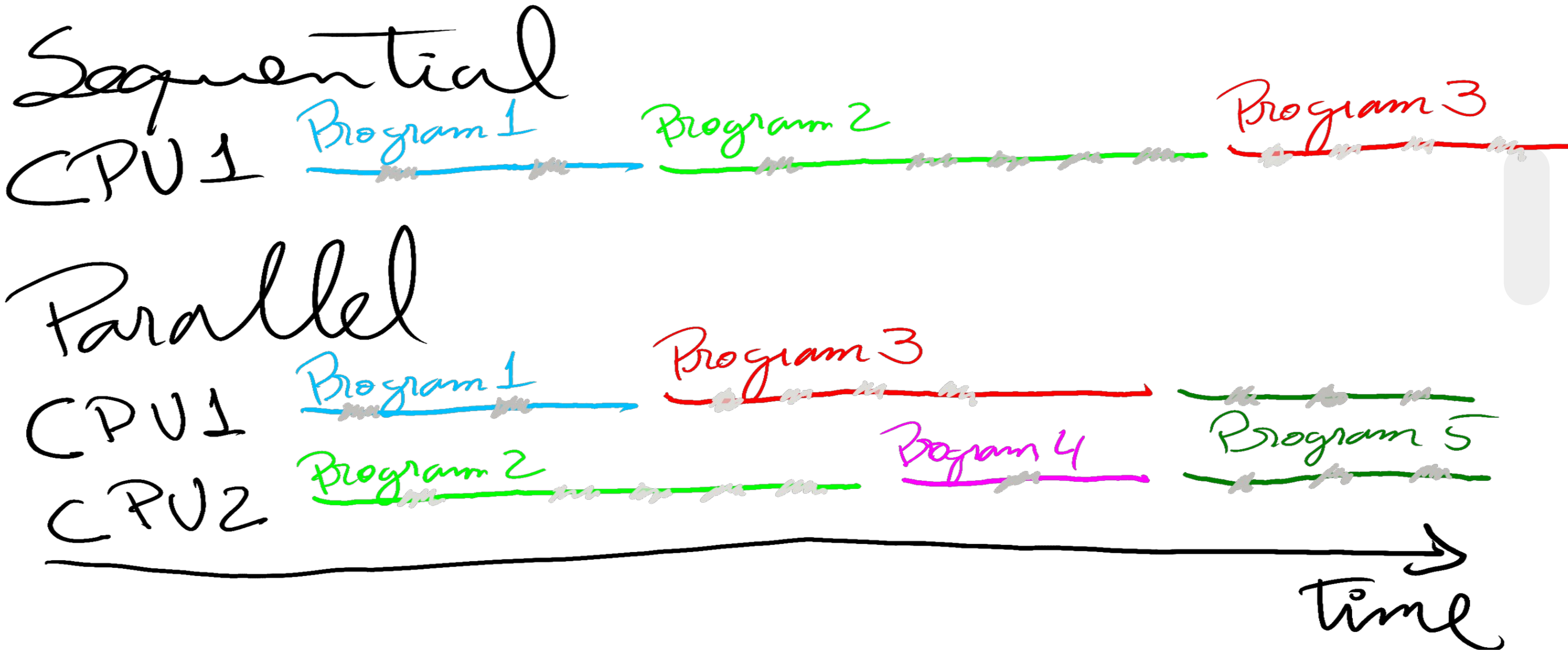
Parallelism vs **Concurrency**

- Mechanism:
  - switch between different tasks
  - work on one task when it can make useful progress
  - suspend one task and work on another thread
- Running the excel, chat and music player at the same time on a single CPU.
  - OS gives each of these programs a small time-slice (~10msec)
- Allow a program to continue execution
  - while waiting for I/O device to respond,
  - letting another thread do useful CPU computation

Concurrency



Parallelism



Concurrency without parallelism

- 1CPU, 1 RAM, 1 I/O
  - While one tasks waits for a device
    - Network I/O, Disk I/O, GPU, ...
  - Other task can be scheduled to use the CPU
  - When data is available on the device
    - Tasks are switched
- Shared resource between tasks
  - CPU or variables

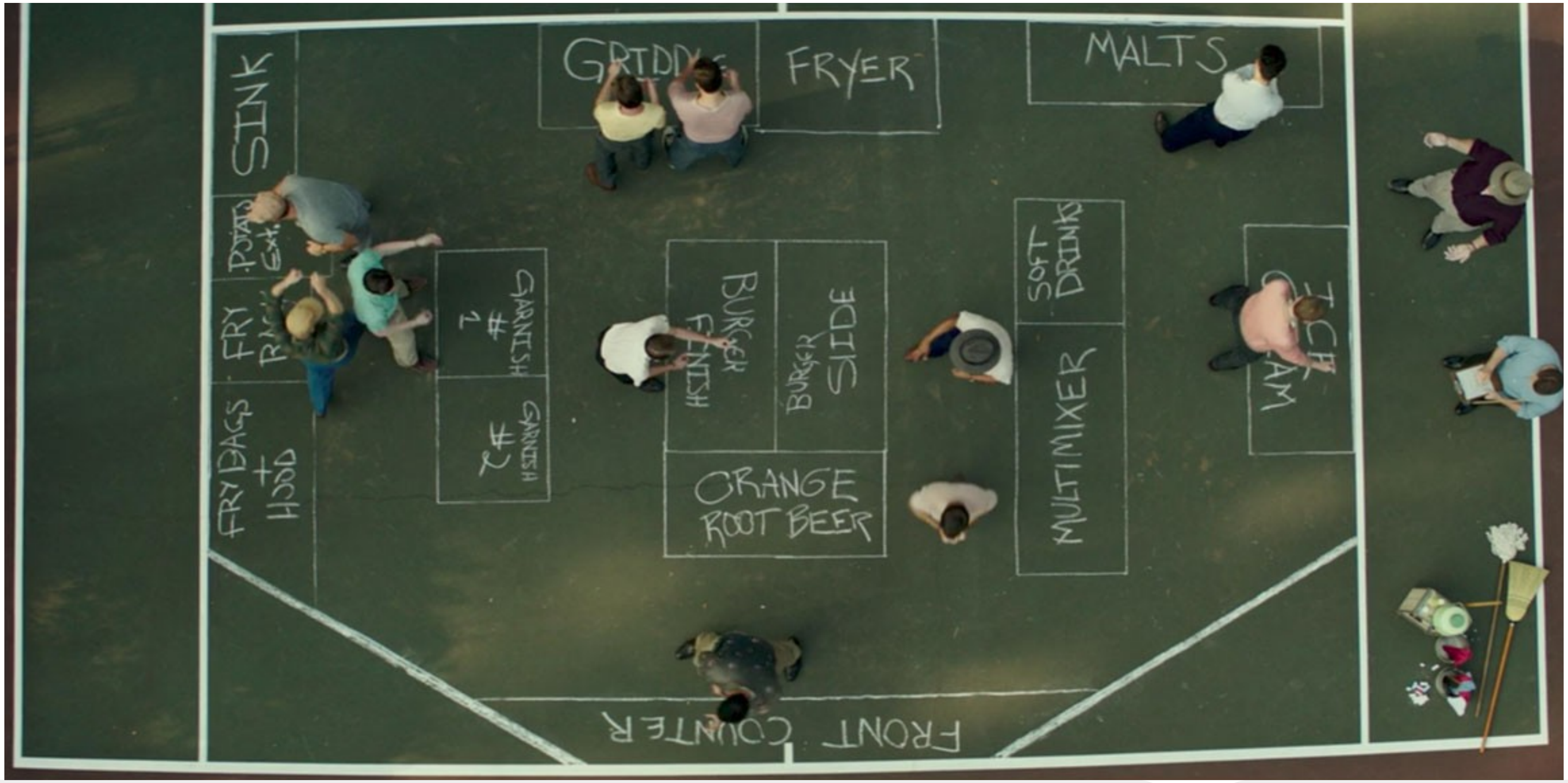
Parallelism without concurrency

- Multiple CPUs/Computers
  - Without sharing resources

Concurrency with parallelism

- PC
  - One vs multiple users
  - One vs multiple programs
  - One vs multiple CPU
- Kitchen
  - One vs multiple cooks
  - One vs multiple knifes
  - One vs multiple recipes

Concurrency with parallelism



Types of parallelism

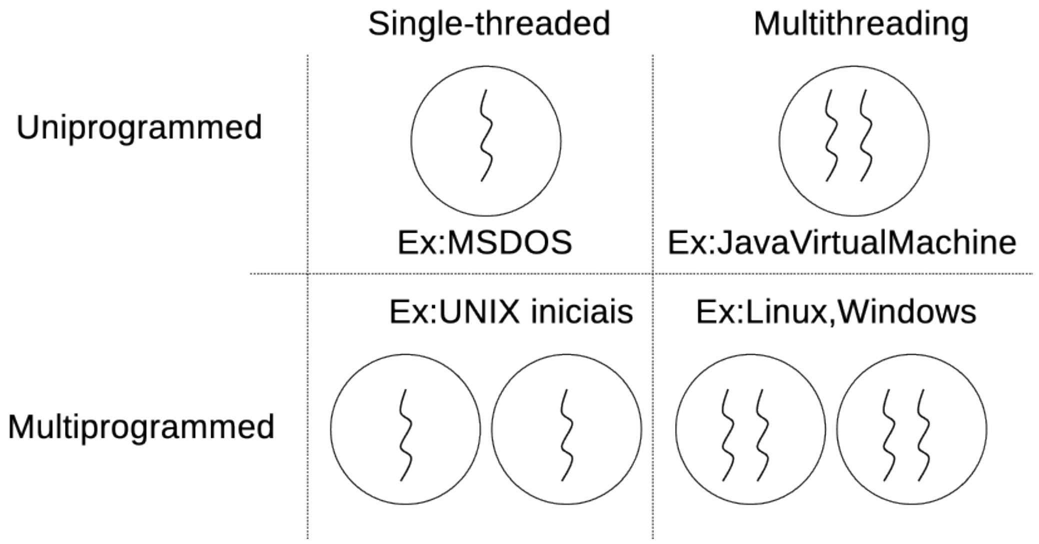
- Data Parallelism
  - same computation being performed on a lot of data
- Task Parallelism
  - different computations/programs running at the same time
- Pipeline Parallelism
  - assembly line:

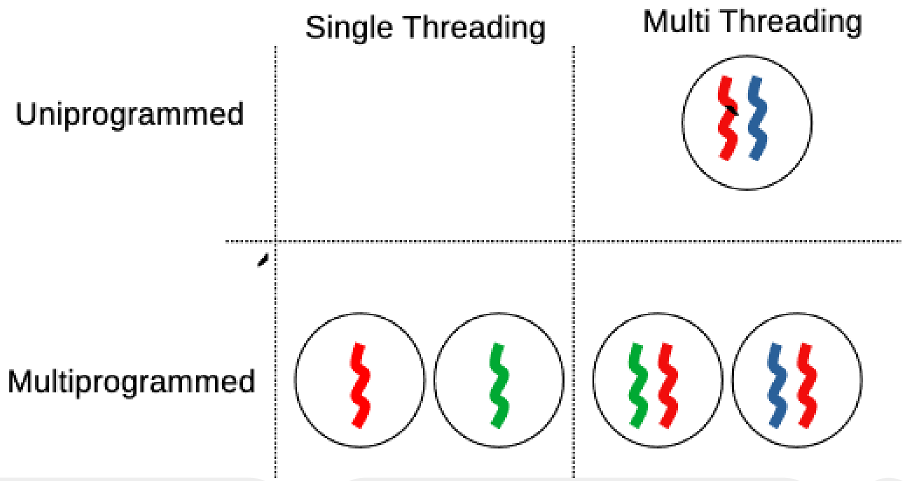
Types of concurrency

- Instruction level
  - the execution of two or more machine instructions simultaneously
- Statement level
  - execution of two or more statements simultaneously
- Unit level
  - execution of two or more subprogram units simultaneously
- Program level
  - execution of two or more programs simultaneously

Concurrency / parallelism in OS

- Processes / Threads
- Single CPU / Multiple CPU



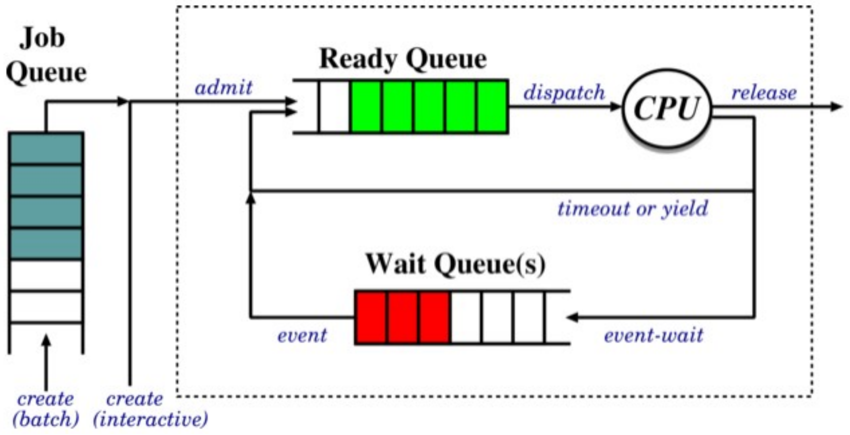


Scheduler

- Selection of the process/task to execute
- Guarantees
  - Fairness - Every process gets its fair share
  - Efficiency - Keep CPU busy 100%
  - Response time – for interactive processes
  - Turnaround – for batch processes
  - Throughput – maximize results

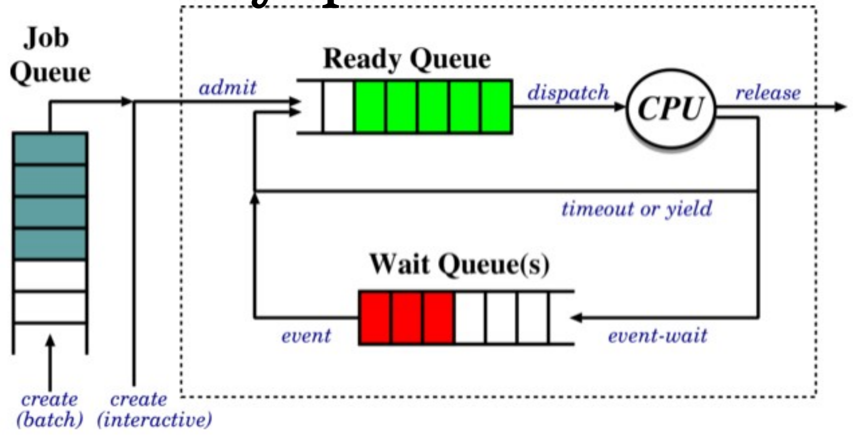
Scheduling concepts

- Queues
  - necessary to store information about tasks/processes
  - Job Queue
    - processes awaiting admission
  - Ready Queue
    - processes in main memory,
      - ready and waiting to execute
  - Wait Queue(s)
    - processes waiting for an IO device (or for other processes)



Scheduling concepts

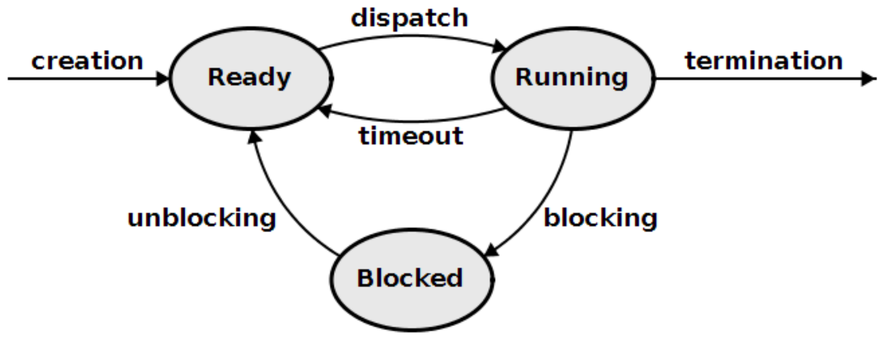
- Job scheduler
  - selects processes to put onto the ready queue



- CPU scheduler
  - selects process to execute next and allocates CPU

Preemptive vs Non preemptive

- When to select the next process to run on the CPU?
- Non-preemptive
  - Running → blocked
  - Running → exit
- Pre-emptive
  - Runing → ready
  - Blocked → ready



Non preemptive

- Simple to implement
  - No timers, process gets the CPU for as long as desired
- Open to denial-of-service:
  - Malicious or buggy process can refuse to yield
- Typically includes an explicit yield call
  - Allow the programmer to release the CPU
- Include implicit yields
  - performing IO, waiting
- Examples: MS-DOS, Windows 3.11

Preemptive

- Solves denial-of-service:
  - OS can simply preempt long-running process
- More complex to implement:
  - Timer management, concurrency issues
- Implemented in modern OS

Scheduling Algoritms

- Round Robin
- Priority (static and dynamic)
- Shortest Job first
- Guaranteed Scheduling
- Earliest deadline first